sPHENIX code development

Michael P. McCumber
sPHENIX Software
5/15/2015

Things we lost in the divorce

sPHENIX universe

GEANT4 sim

PHENIX universe

Fast sim

Jet output

Jet background subtraction

Flow Jets

much of this was user-land code with non-standard outputs

We need to port this back into sPHENIX, but it is best if we redesign it so that it plays well with the GEANT4 interface

Work Plan:

† - good match to me

*** - others can help**

Jet Storage† - a default set of jet reconstructions for common use (will be completed today)

Jet Reconstruction Code* - port the ATLAS method (???) and our Particle Flow reco (Javier)

Evaluation Objects† - break the SVTX and Calo evaluation into two modules:
(1) places new association contains onto the DST (SVTXEVAL/SvtxTrack_MCParticle_MultiMap)
(2) reads the objects and dumps out the ntuple files

FastSim†* - modules to read the G4 input list, produces smeared tracks, simulates towers, and eval lookup objects, swappable with GEANT4 modules

//—Restored functionality from divorce————————————————————

SVTX reco improvement†* - understand increase in 6+1 track reconstructions, fix?

Jet Evaluator† - traces the ancestry to report true jet momentum, etc. Outputs to DST objects and new ntuple

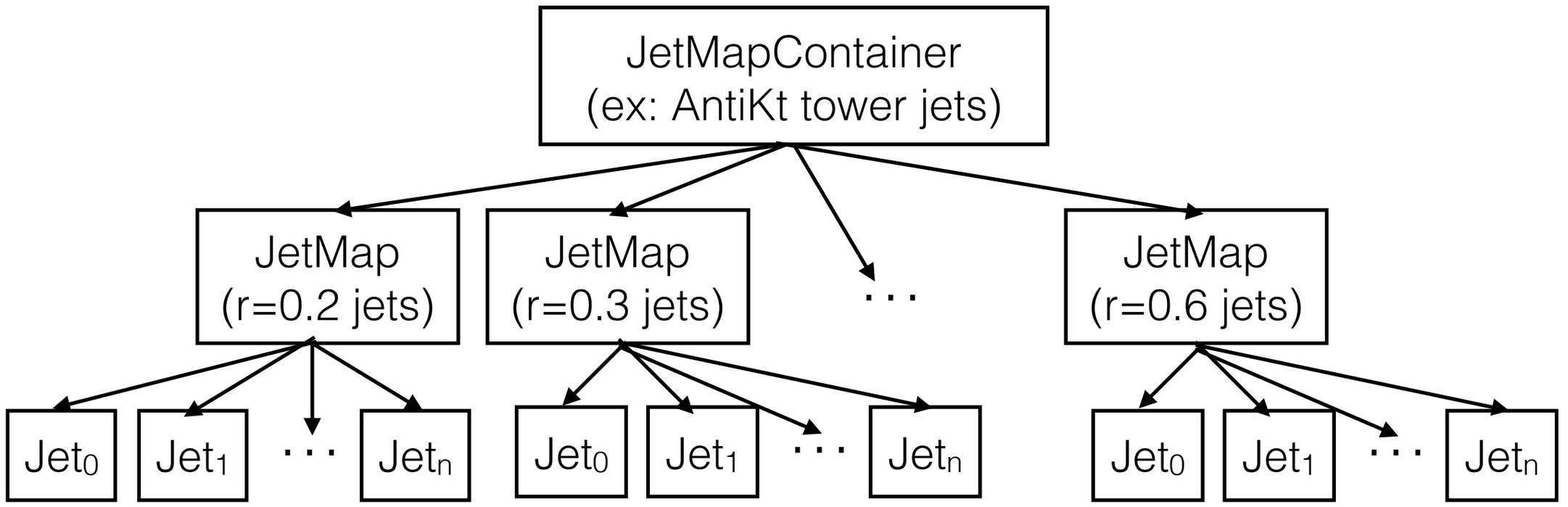
SVTX+TPC evaluation†* - what does the TPC need to produce a proper evaluation and output object?

SVTX Track Redesign†* - very wasteful storage-wise, doesn't contain a full set of track projections and covariance, prevents modular reconstructions

SVTX Reco Modularization* - vertex reconstructions with Rave, separate Kalman fitter with full GEANT4 material, full Kalman fit with vertex -> SvtxPrimaryTrack (for DCA-less studies)

Jet Reconstruction Storage

Plan A: A Map of Map of Jets



Storage on the DST would appear:
Jets/
  TowerAntiKtJets
  CaloAntiKtJets
  FlowAntiKtKets

Usage Scenarios:

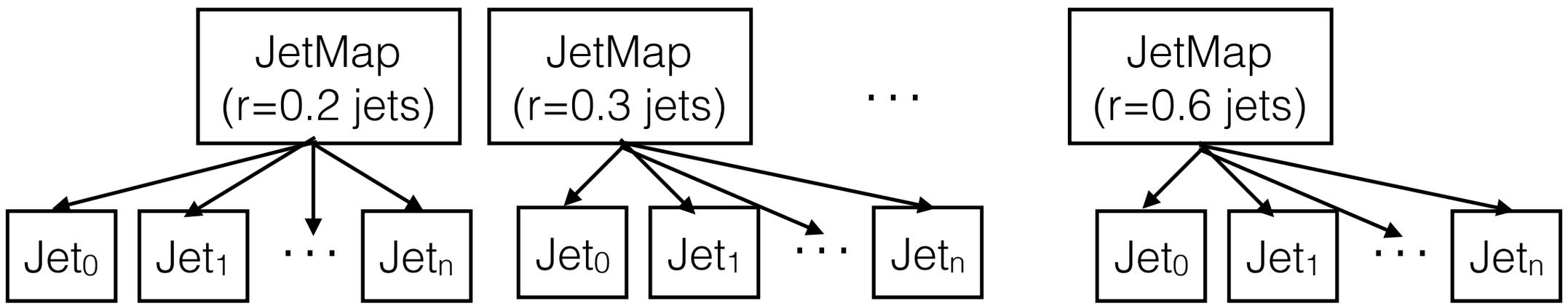
Loop over JMC->begin; JMC->end
Loop over JM->begin; JM->end
Jet* = &iter->second

Loop over JM[r]->begin; JM[r]->end
Jet* = &iter->second

Advantage that one can either loop over all Tower AntiKt jets at once, jet radii aren't stored by name, storages sorts by typical use scenario
Disadvantage of two calls to get directly to a jet (what radii, what id)

Jet Reconstruction Storage

Plan B: A Map of Jets



Storage on the DST would appear:

Jets/
- TowerAntiKtJetsR0p2
- TowerAntiKtJetsR0p3
- …
- TowerAntiKtJetsR0p6
- CaloAntiKtJetsR0p2
- …
- CaloAntiKtJetsR0p6
- FlowAntiKtKetsR0p2
- …

Usage Scenarios:

Go out and fetch each node separately

```
Loop over JM->begin; JM->end
  Jet* = &iter->second
```

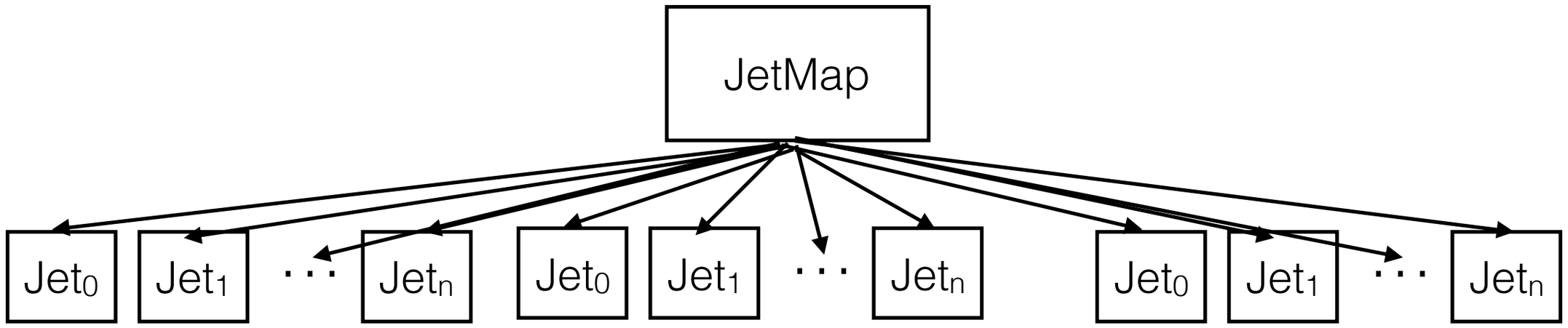
```
Loop over JM->begin; JM->end
  Jet* = &iter->second
```

Advantage - Simple, single place for map-wise info, familiar to users

Disadvantage - Little messy, difficult to loop over all jets from a particular algorithm without already knowing all R values at compile time

Jet Reconstruction Storage

Plan C: A Map of Jets



Storage on the DST would appear:
Jets/
  Jets

Usage Scenarios:

develop a search that returns the algorithm and algorithm parameter as a map of pointers

map<Jet*> jets = JM->fetch("AntiKtCaloJets",R);

Advantage - flexible for future algorithms and parameters, needed?
Disadvantage - Searching repeatedly will be slow, obscured storage of what is available

Generally seems like a bad idea…

Jet Storage Object

Plan A: Lightweight 4-vector + associations

Stores:
- unique identifier within the map
- 4 vector: px, py, pz, energy (alternate: p, eta, phi, et)
- associations to constituent object unique ids (towers, clusters, tracks)
  - multimap< sourceid, unsigned int>
- under consideration: 4x4 covariance uncertainty (px,py,pz,e)

Interface:
- gets/sets for px,py,pz,e
- handles for pt, ET, p, phi, eta, etc
- bring multi map interface out for ancestry tracing operations

Plan B: Extend PseudoJet/PHJet/TLorentzVector object

Inherits from PseudoJet/PHJet/TLorentzVector
Adds multi map storage for clustered objects

Interface:
- PsuedoJet interface
- bring multi map interface out for ancestry tracing operations

Jet Reconstruction Library

We need a single module for each “kind” of jet reconstruction:

TowerAntiKtJetReco
CaloAntiKtJetReco
ParticleFlowAntiKtJetReco
} (need to include a port of the ATLAS bkg method)
(other algorithms?)
Jet
JetMap
JetMapContainer (plan A only)

I’ve started a private g4jets library based on Plan A options, filled in storage objects, compiles.

Pick a plan, I will modify to suit, and submit into github today.

I’ll start on Eval objects when I get back.

Future Jet Studies

Particle Flow Jets - better calorimeter clustering

Bottom Jets
- initial b-jet id in p+p with G4
- tracking: what design is needed for 2nd vertex reco

Jet Rejection - shape cuts

Jet Purity Studies with Jet Modification

Recoil jet background subtractions

Raw level tune of the FastSim